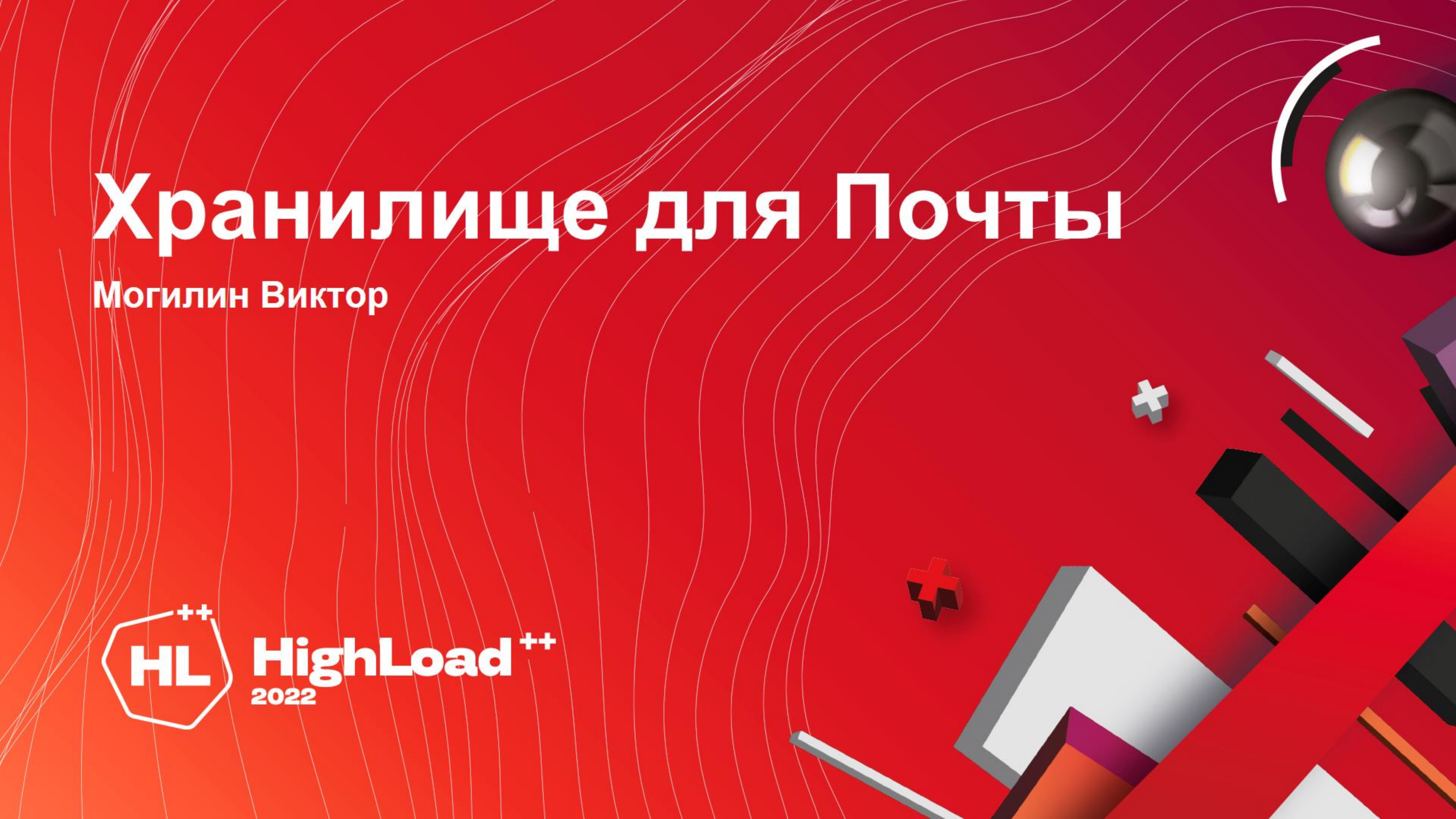
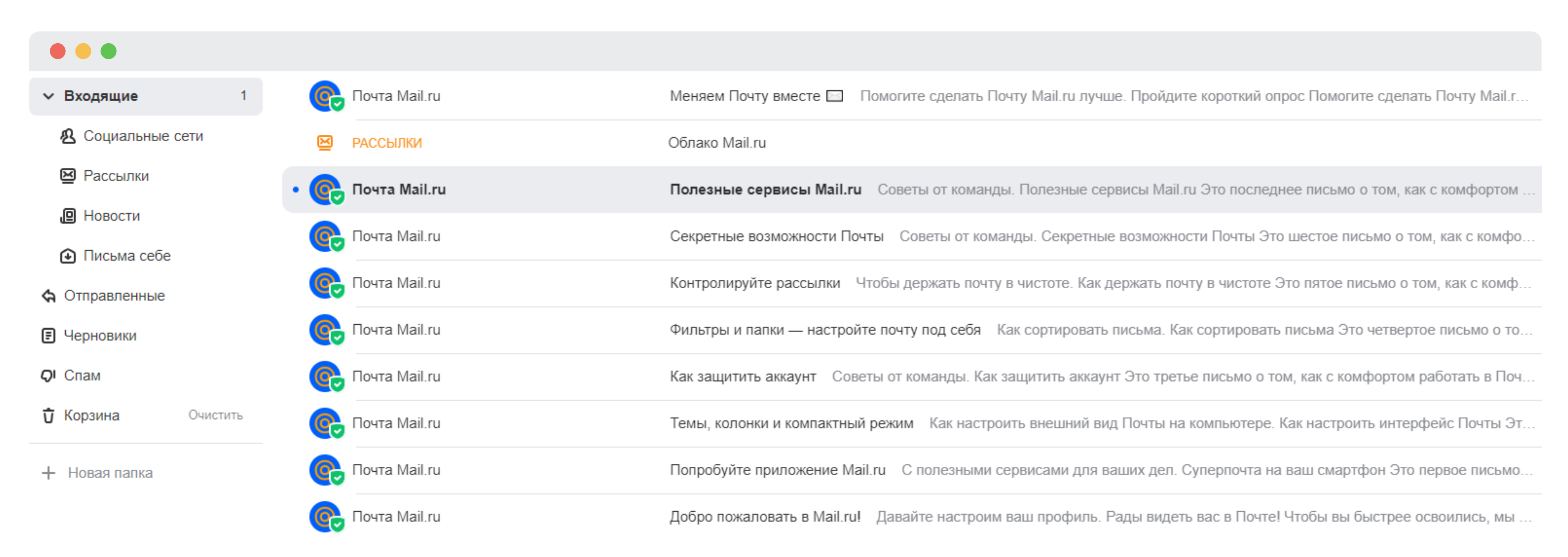

Хранилище для Почты
Могилин Виктор
HL
HighLoad++
2022



Хранилище для Почты



Могилин Виктор
Руководитель группы разработки стораджей в Почте Mail.ru
v.mogilin@corp.mail.ru


Входящие 1
Социальные сети
Рассылки
Новости
Письма себе
Отправленные
Черновики
Спам
Корзина
Очистить
Новая папка
Почта Mail.ru
Меняем Почту вместе
Помогите сделать Почту Mail.ru лучше. Пройдите короткий опрос Помогите сделать Почту Mail.r...
РАССЫЛКИ
Облако Mail.ru
Почта Mail.ru
Полезные сервисы Mail.ru
Советы от команды. Полезные сервисы Mail.ru Это последнее письмо о том, как с комфортом ...
Почта Mail.ru
Секретные возможности Почты
Советы от команды. Секретные возможности Почты Это шестое письмо о том, как с комфо...
Почта Mail.ru
Контролируйте рассылки
Чтобы держать почту в чистоте. Как держать почту в чистоте Это пятое письмо о том, как с комф...
Почта Mail.ru
Фильтры и папки — настройте почту под себя
Как сортировать письма. Как сортировать письма Это четвертое письмо о то...
Почта Mail.ru
Как защитить аккаунт
Советы от команды. Как защитить аккаунт Это третье письмо о том, как с комфортом работать в Поч...
Почта Mail.ru
Темы, колонки и компактный режим
Как настроить внешний вид Почты на компьютере. Как настроить интерфейс Почты Эт...
Почта Mail.ru
Попробуйте приложение Mail.ru
С полезными сервисами для ваших дел. Суперпочта на ваш смартфон Это первое письмо...
Почта Mail.ru
Добро пожаловать в Mail.ru!
Давайте настроим ваш профиль. Рады видеть вас в Почте! Чтобы вы быстрее освоились, мы ...

```
From: <v.mogilin@corp.mail.ru>
To: <v.mogilin@corp.mail.ru>
Subject: Hello
Date: Tue, 25 Jan 2022 23:04:21 +0300

Hello world
```

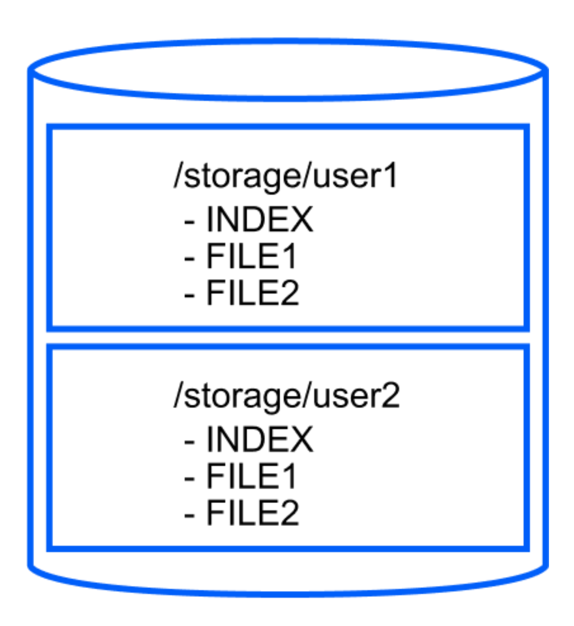
*стандарты rfc-822, rfc-2822, rfc-5322

Проблема: вертикальное масштабирование

Вложения: **3TB**

Письма + индексы: **1TB**

Предел: диски **4TB**



long long ago…

IOPS

Диски растут по объему, но не растут по IOPS   :(

Random IO:
 - обновление индексов, запись новых файлов

Чтение:
 - directory metadata + load inode + read file
 - фрагментация, т.к. есть удаления

*IOPS – количество элементарных операций ввода-вывода в секунду

Задача: на порядок сократить число серверов

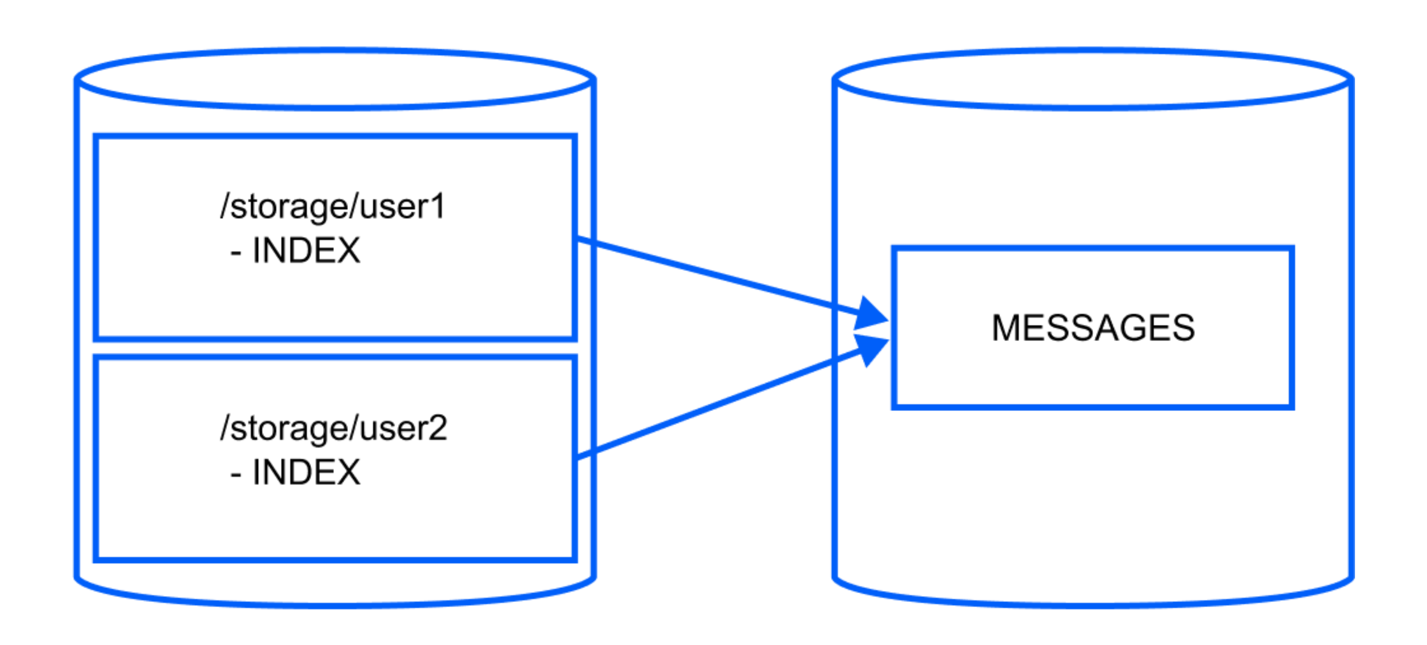
~3К → ~300

*маленькие диски – много серверов

	Volume	RPS	Immutable
Meta	2PB	3M	нет
MESSAGE	15 PB	80K	да
Attach	50PB → 35PB	8K	да

*задача по хранению почтовых вложений – доклад от Андрея Сумина

Решение: выносим письмо на другой диск



HDD vs NVME: 15PB данных

	Всего	Примечание	Количество серверов
NVME	$2.5M	8Tb NVME - $1200	150
HDD	$450K	18Tb HDD - $500	12

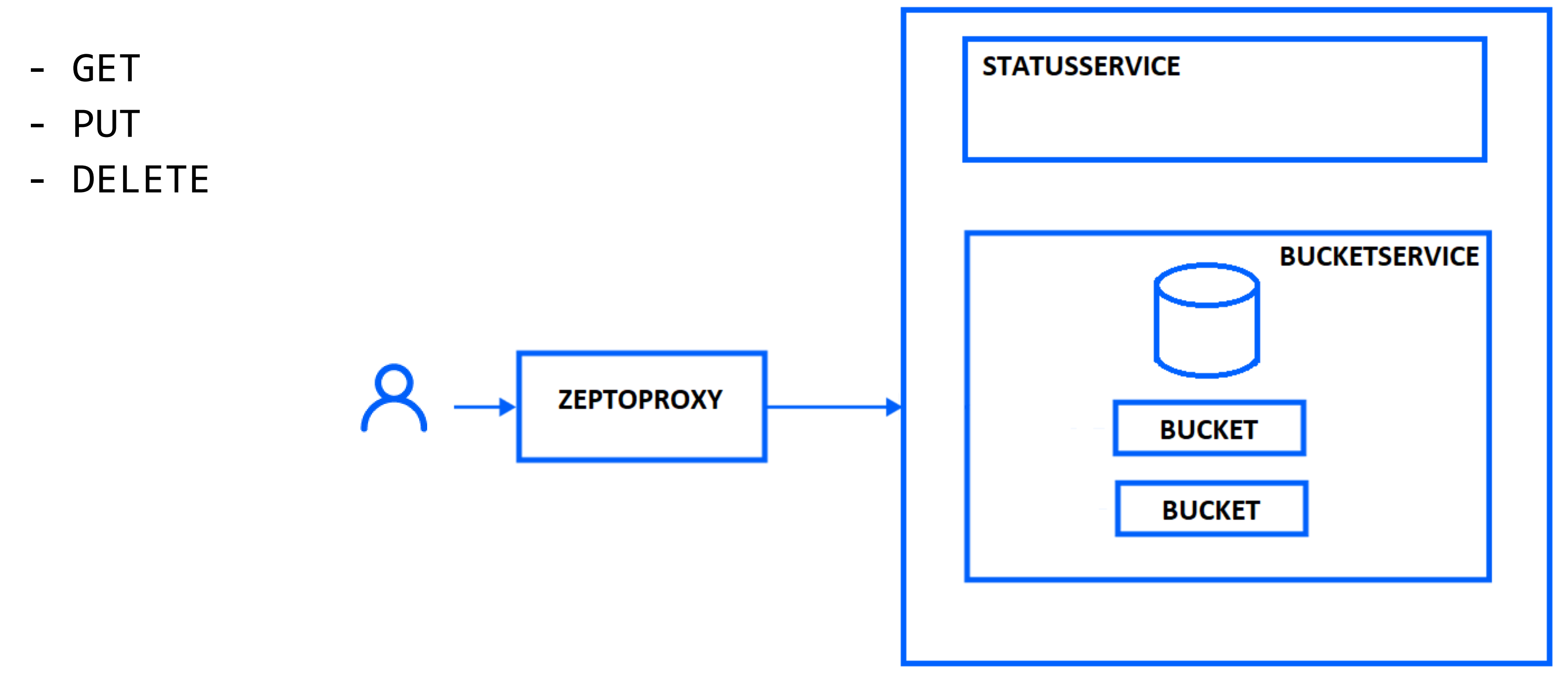
1 сервер: – 72 диска (12 + SATA-полка на 60 дисков)

Стратегия

- Письмо – не модифицируется
- Письмо – это BLOB!

Объектное хранилище – минимальный IO для доступа к данным

zepto: собственное BLOB-хранилище



*bucket – он же image, он же volume

bucket: физический уровень

- **bucket** – это файл
- append only
- fallocate (2GB)

fallocate:

- непрерывная последовательность блоков
- гарантирует, что место не закончится «посередине записи»

bucket

HEADER	RECORD 1	RECORD 2	RECORD 3	EOF

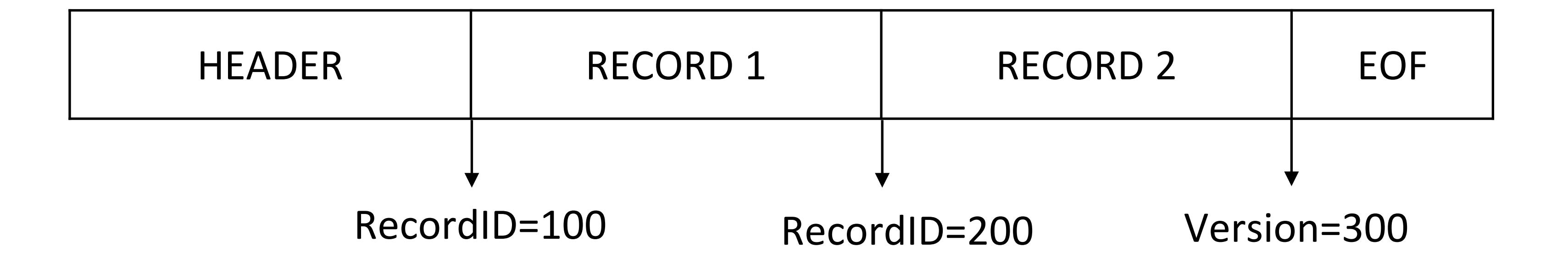
```
type BucketFile struct {
    compactVersion   timestamp
    lastRewrite      timestamp
}
```

record

HEADER	PAGE 1 4Kb	PAGE 2 4Kb	PAGE 3 4Kb

```
type RecordHeader struct {
    RecordID     uint32
    Size         uint32
    DataSize     uint32                          // for compressed data
    Type         RecordType                      // ADD, DEL, EOF, …
    Encoding     int8                            // zstd
}
```

bucket: RecordID и версия



*бакет «закрывается» на запись, когда его version > 2GB

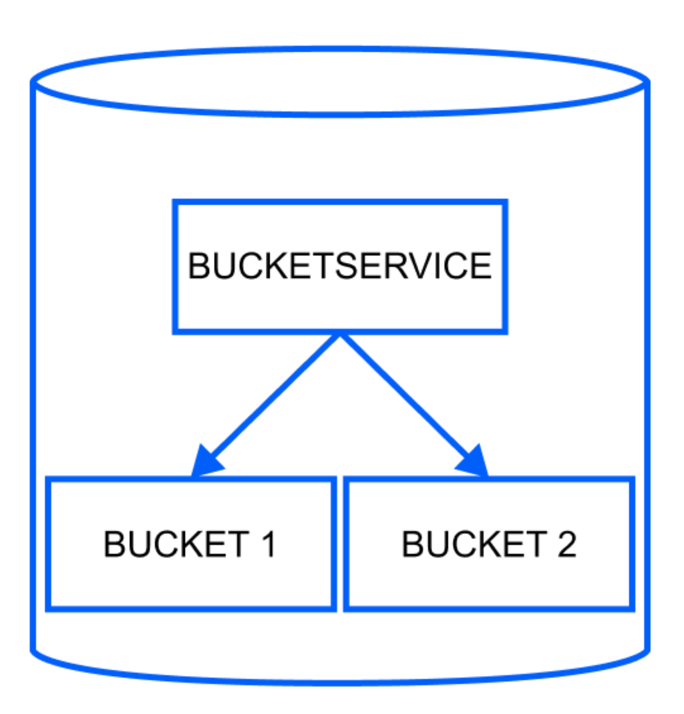
page

DATA	CRC
4092	4

`*проверяем crc на каждом чтении и периодически`

bucketservice

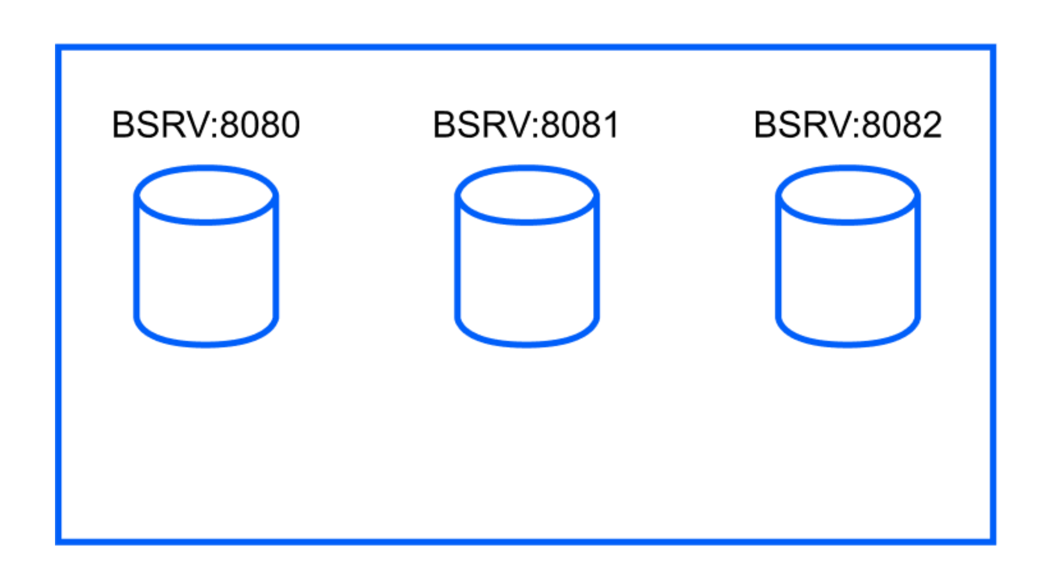
- read
- write



bucketservice

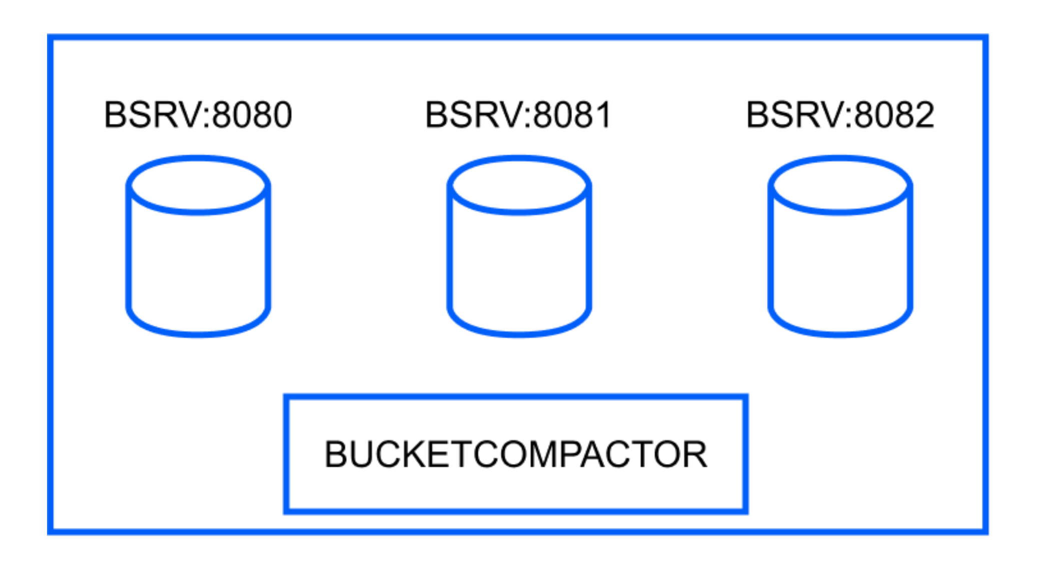
1 = **1**

диск bucketservice

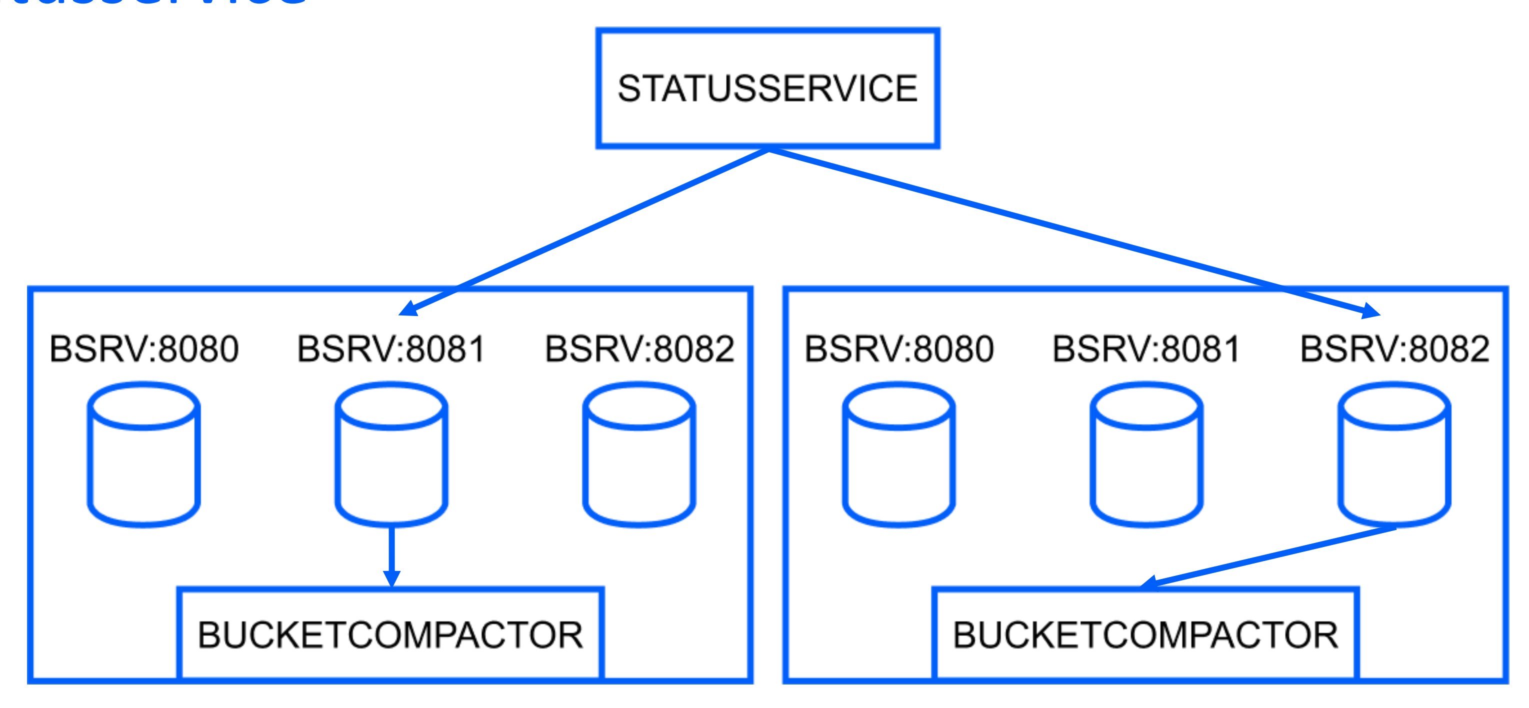


bucketcompactor

- crc check
- создание бакетов



statusservice



statusservice: in-memory-индекс бакетов

```
bucket1={
    version: 100,
    address: ip:port,
    status:  healthy
},

bucket2={
    version: 200,
    address: ip:port,
    status:  broken
}
```

statusservice

API:
- найти bucket по ID
- выдать bucket для записи

Maintenance:
- поддерживает в кластере >= N доступных writable-бакетов
- планирует сверки контрольных сумм

IOPS: за счет чего уменьшается IO

Размер бакета 2GB:

- уменьшается metadata overhead (мало файлов на диске)
- ext4/xfs – оптимизации на больших файлах (extent tree)

fallocate:

- меньше проблем с фрагментацией (блоки последовательно)

Итого:

- write – (в идеале) 100МБ/сек
- read – один IO

Надежность хранения

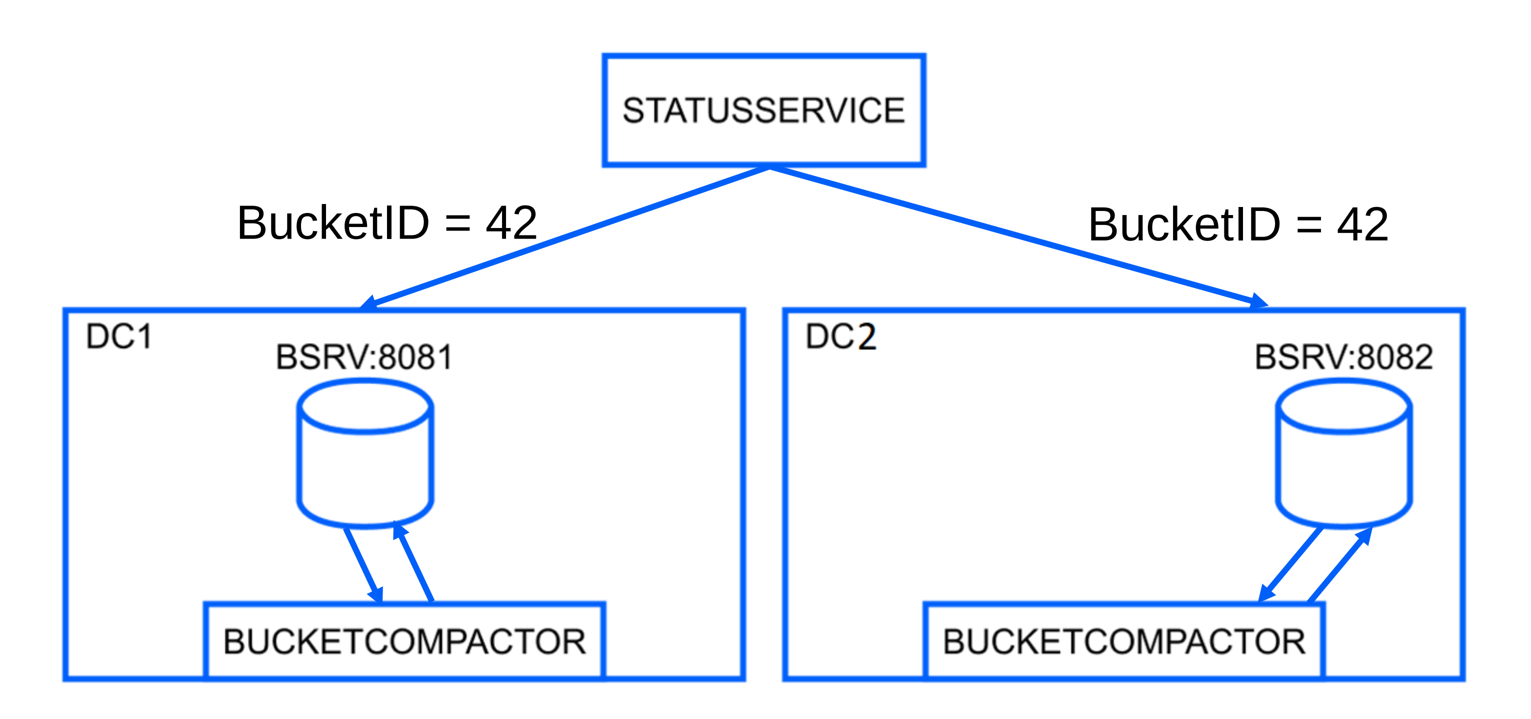
Способность не терять данные в случае выхода из строя диска

Репликация

```
Физический уровень (bucketservice):
 - файл
 - EOF


Логический уровень (statusservice):
 - BucketID
 - набор реплик для бакета
```

Создание bucket: replication factor x2



statusservice: индекс бакетов

```
42={
    version: 0,
    address: [ip1:8081, ip2:8082],
    status:  healthy
}
```

zeptoproxy при запросе бакета на запись получает набор адресов

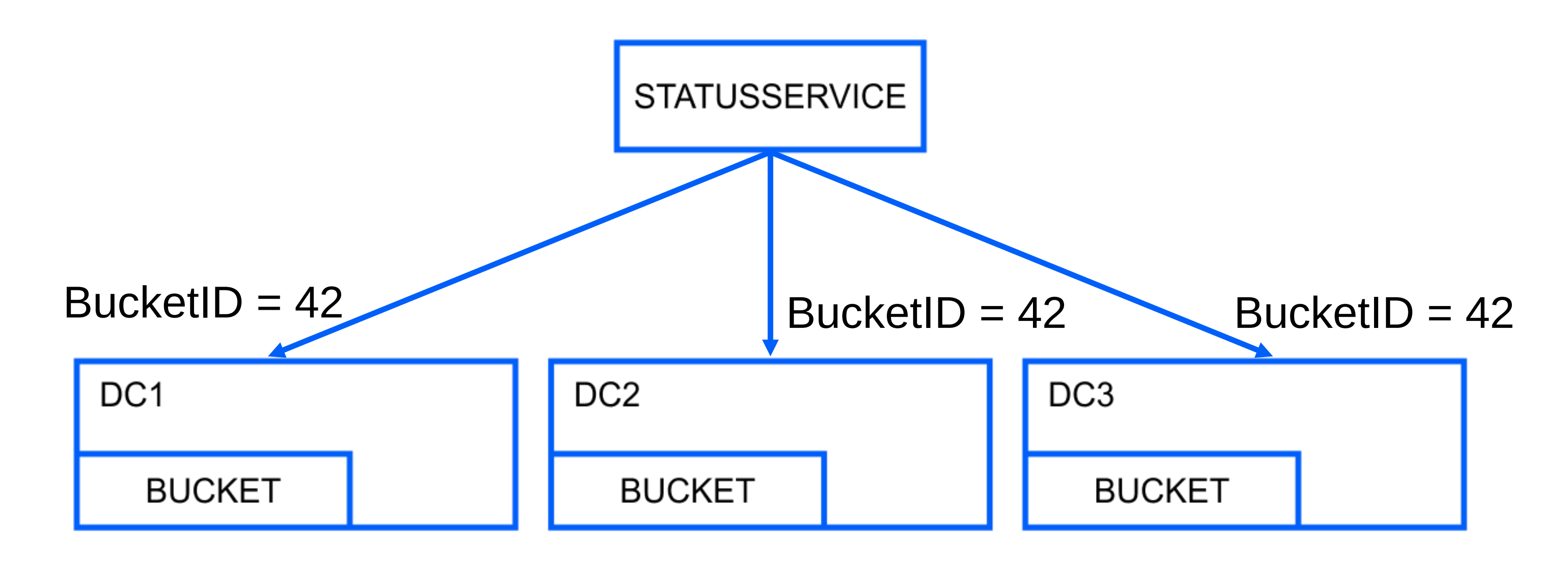
replication factor

x2

x3

x1.5

replication factor x1.5



*erasure coding

replication factor x1.5



*«полторирование»

Надежность

- репликация
- crc
- fix “сломанных” бакетов

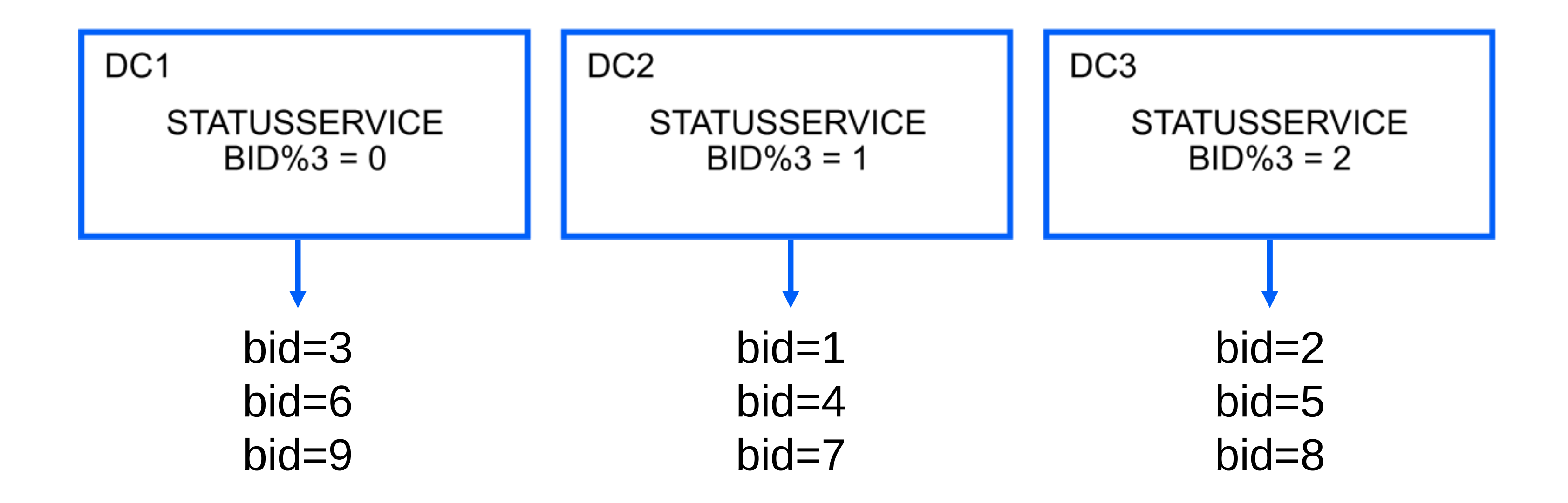
Доступность

Возможность чтения/записи при потере DC

2 уровня:

- statusservice
- бакеты

statusservice: схема шардирования



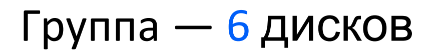
*master для «своих» бакетов

statusservice schema

	DC1	DC2	DC3
group1	IP1:port1	IP2:port2	
group1	IP3:port3		IP4:port4
group1		IP5:port5	IP6:port6

*админы добавляют группами

x2 в 3 DC



Группа — 6 дисков

DC1
DISK 1
DISK 2
DC2
DISK 3
DISK 4
DC3
DISK 5
DISK 6

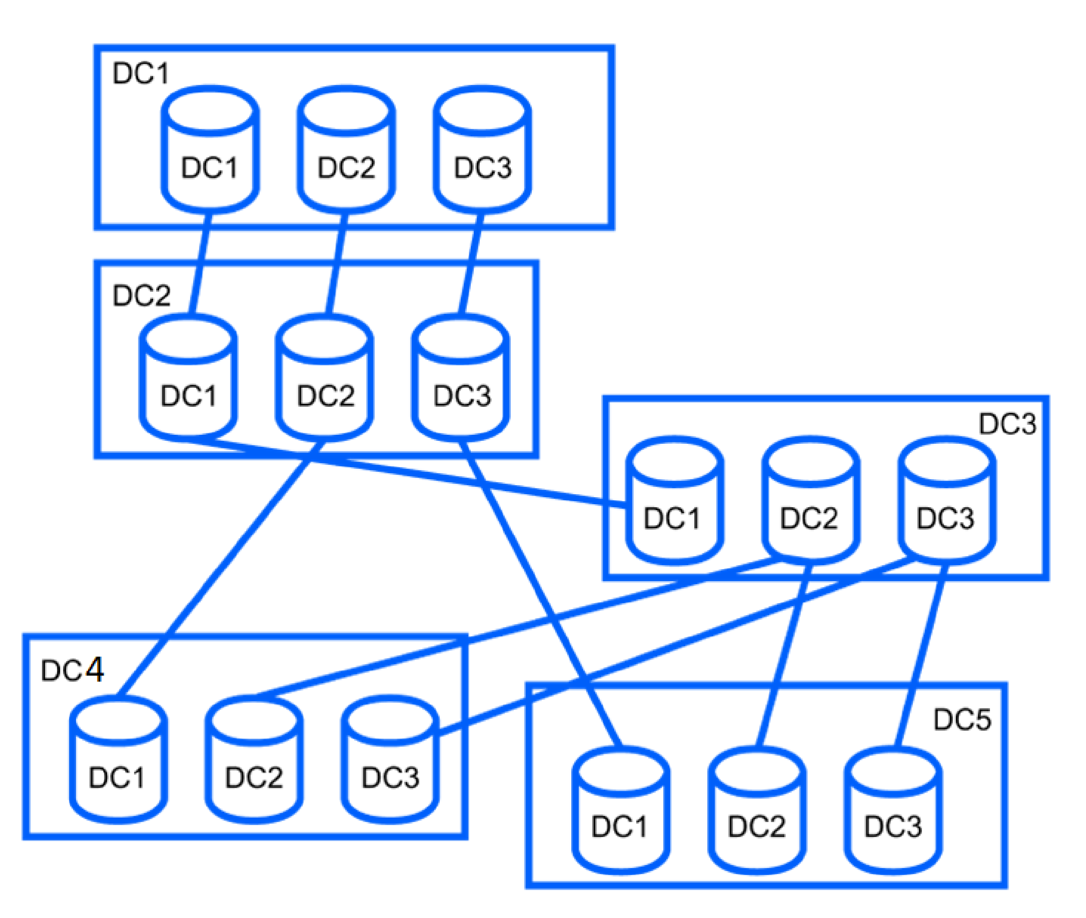
x2 в 3 DC

1/3 доступно на запись

DC1
DISK 1
DISK 2
DC2
DISK 3
DISK 4
DC3
DISK 5
DISK 6

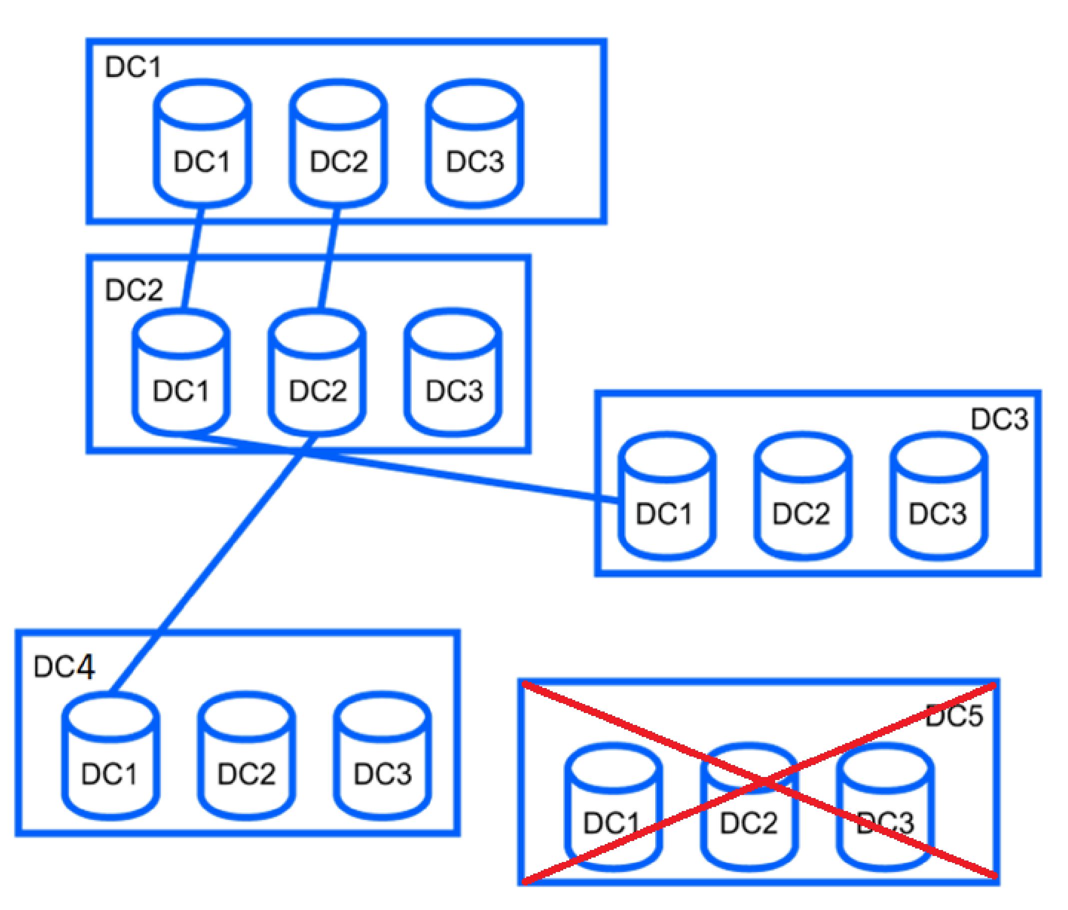
x3 в 5 DC

Группа – 15 дисков

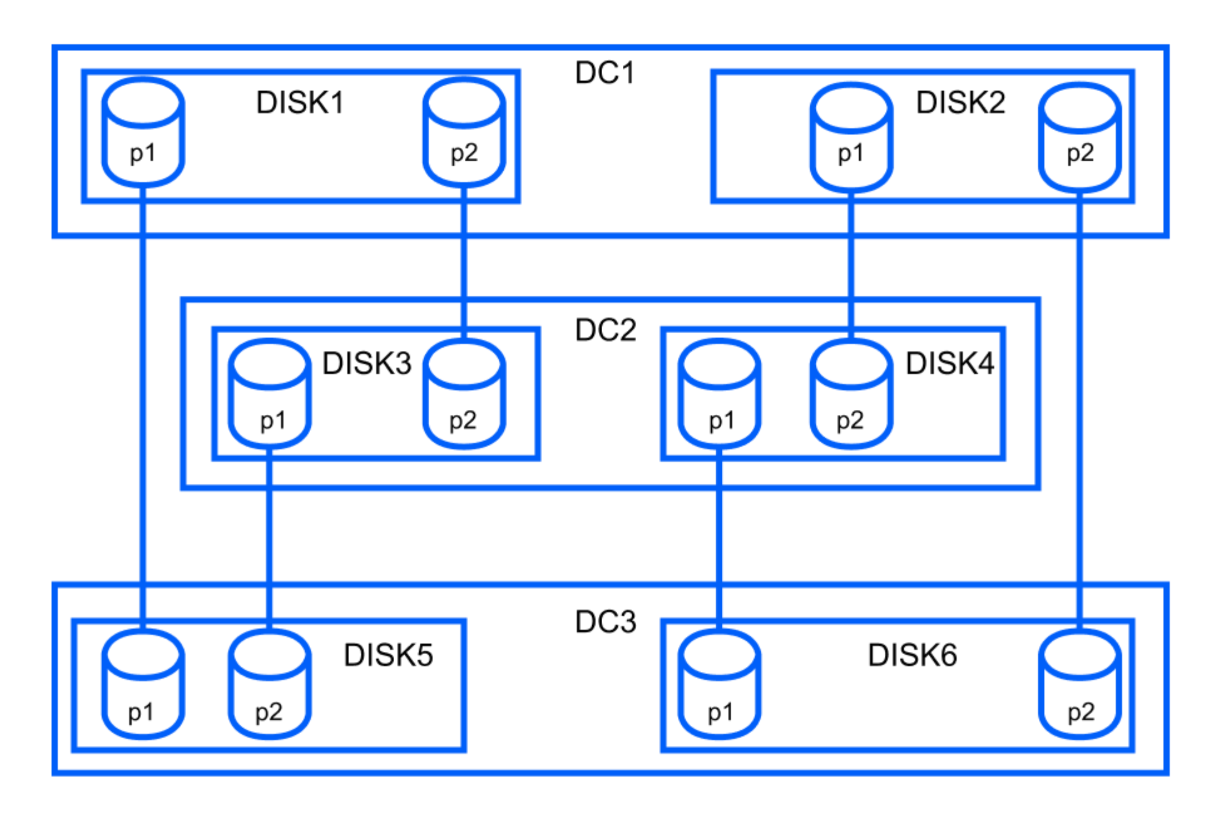


x3 в 5 DC.

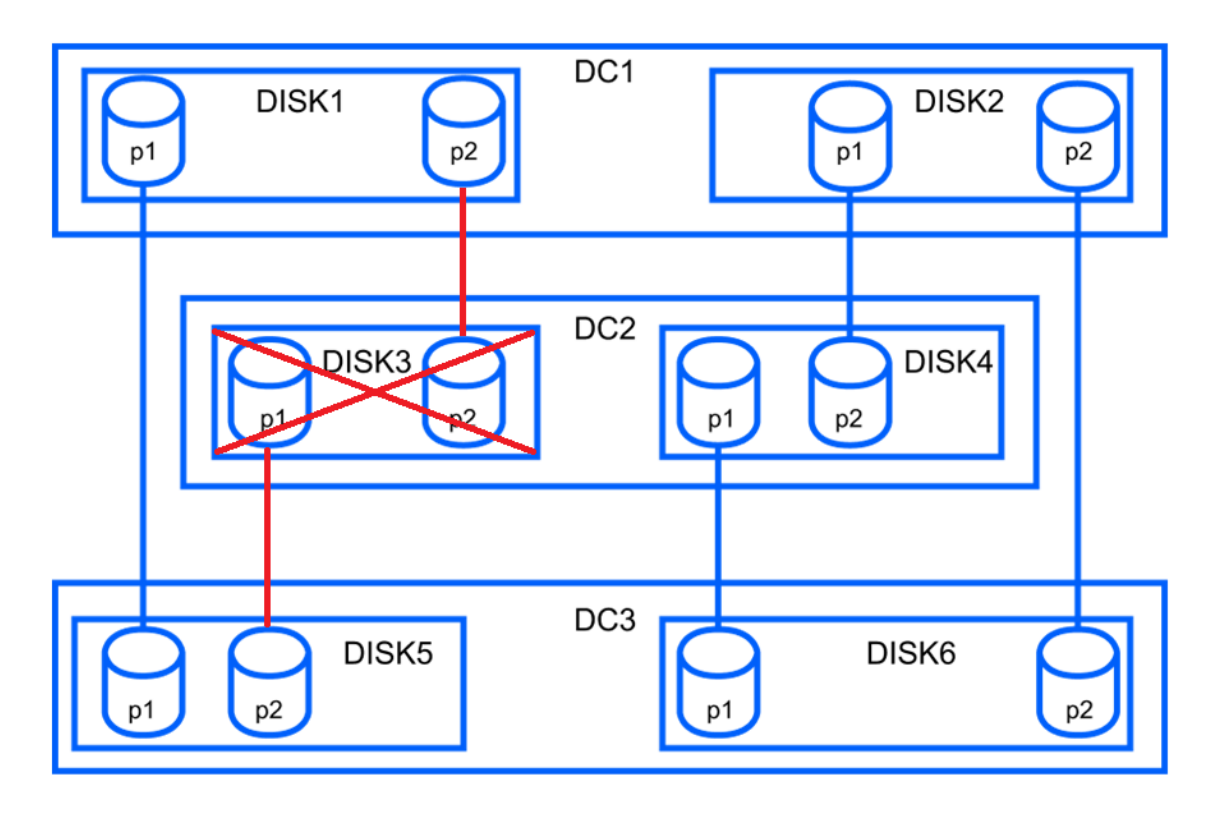
- 2/5 доступно на запись



Пары – это logical partitions

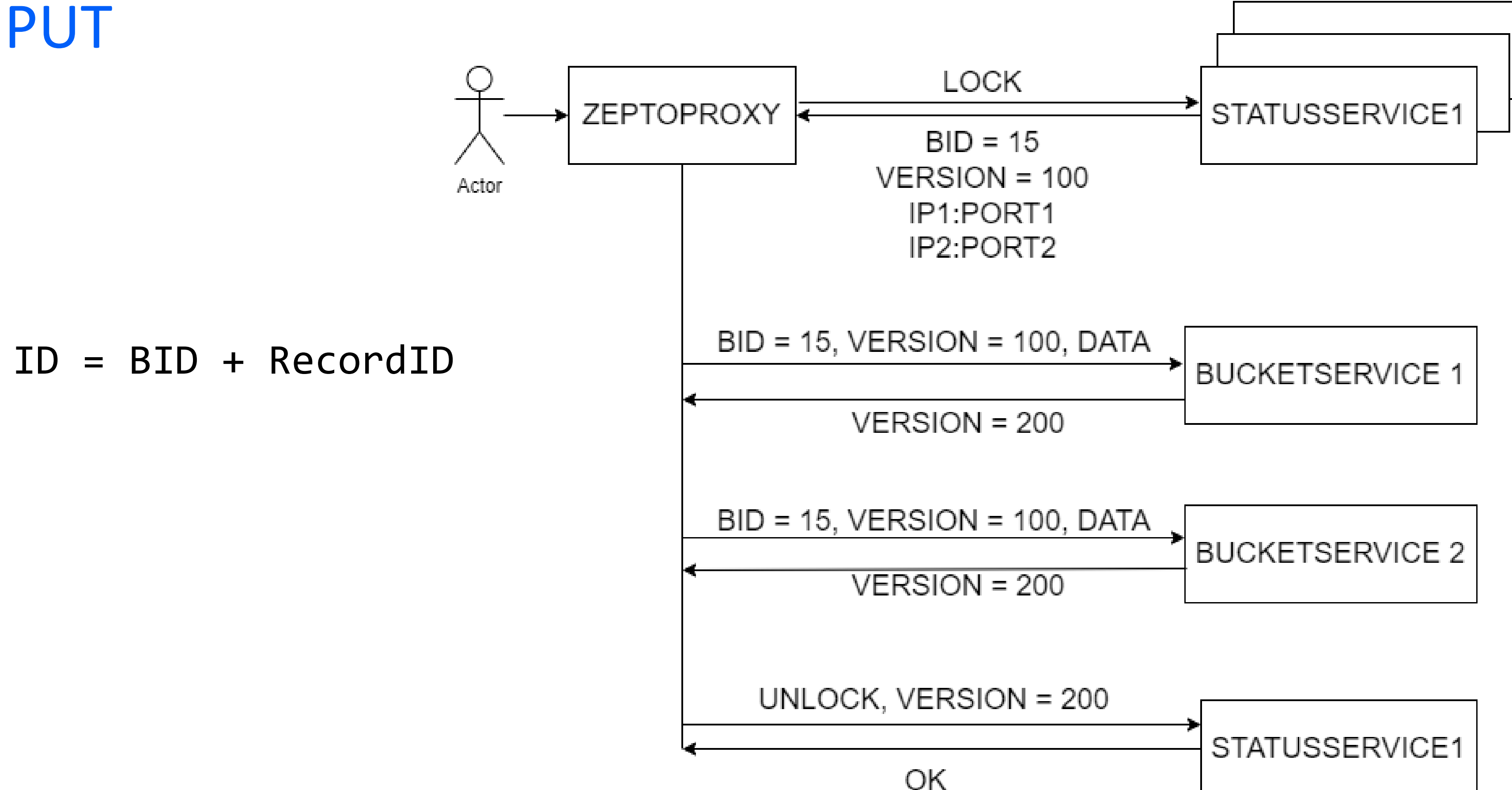


Пары – это logical partitions



PUT

ID = BID + RecordID



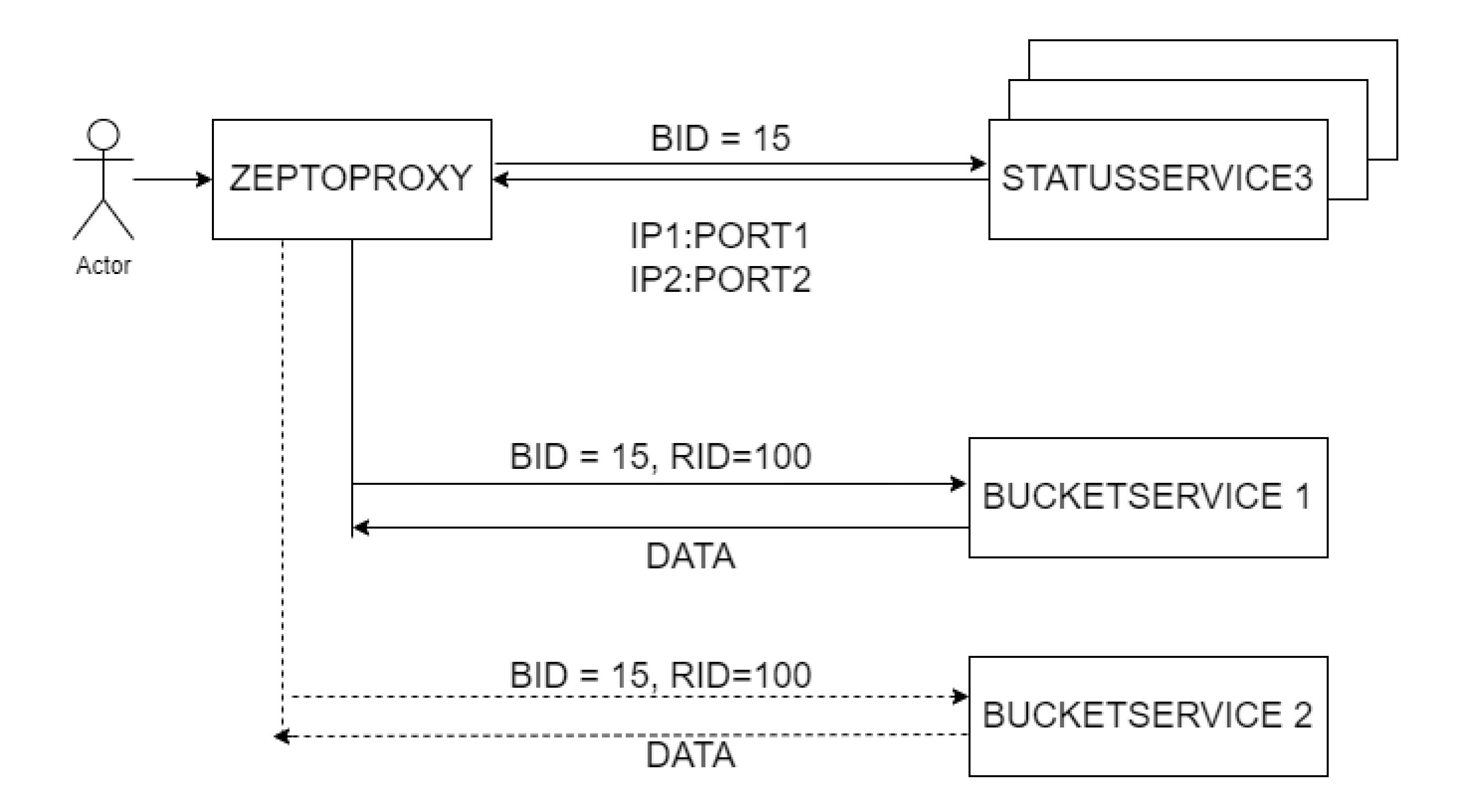
zepto: ограничение by design

ID = BID + RecordID, т.е. **2^64 байт (16 эксабайт)**

всего в почте **15PB**, так что пока не переживаем ;)

Pros:
- ZeptoID – число, которое можно кодировать
- не нужны индексы

GET



Actor
ZEPTOPROXY
BID = 15
IP1:PORT1
IP2:PORT2
STATUSSERVICE3
BID = 15, RID=100
BUCKETSERVICE 1
DATA
BID = 15, RID=100
BUCKETSERVICE 2
DATA

DELETE

HEADER	RECORD 1	RECORD 2	RECORD 3	DEL 1	DEL 2	EOF

- записывается в конец даже закрытого бакета
- lock на конкретный bucket

DELETE: после компакта

HEADER	DEL INDEX	RECORD 3	EOF

DEL INDEX: rid ⟶ offset

*могут быть foreign delete’ы

Компакты

statusservice запускает компакт:
- превышен порог удалений (получает от bucketservice)
- по времени (в том числе сверка crc)

Доступность

- построение кластера в соответствии со схемой в разных DC
- ретраи (в том числе с перебором DC)
- foreign delete

zeptoctl: управление кластером

```
 - схема групп для statusservice
 - схема шардирования statusservice


zeptoctl move in  bsrv1,bsrv2:
 - выровнять нагрузку на новых серверах
 - масштабирование



*disk_usage по кластеру = total/used на каждом bucketservice
```

zeptoctl buckets

	bucket					bucket stats			daemon info	
group	bid	status	writable	version	type	compact	seen	dels	dc	directory
group1	1	broken	false	4624	reg	11h3m59s	32s	0		/tmp/zepto/disk1
group1	1	ok	false	4624	reg	1m37s	32s	0		/tmp/zepto/disk2
group1	2	ok	true	4252	reg	1m34s	32s	0		/tmp/zepto/disk3
group1	2	ok	true	4252	reg	1m37s	32s	0		/tmp/zepto/disk4
group1	3	ok	true	0	reg	1m37s	32s	0		/tmp/zepto/disk3
group1	3	ok	true	0	reg	1m34s	32s	0		/tmp/zepto/disk4
group1	4	broken	false	0	reg	11h4m0s	32s	0		/tmp/zepto/disk1
group1	4	ok	false	0	reg	1m34s	32s	0		/tmp/zepto/disk2

zeptoctl schedule --fix

	bucket					bucket stats			daemon info		
group	bid	status	writable	version	type	compact	seen	dels	dc	directory	free
group1	1	ok	true	4624	reg	20.5s	20.5s	0		/tmp/zepto/disk1	29GB
group1	1	ok	true	4624	reg	21.5s	21.5s	0		/tmp/zepto/disk2	29GB
group1	2	ok	true	4252	reg	21.5s	21.5s	0		/tmp/zepto/disk3	29GB
group1	2	ok	true	4252	reg	20.5s	20.5s	0		/tmp/zepto/disk4	29GB
group1	3	ok	true	0	reg	20.5s	20.5s	0		/tmp/zepto/disk3	29GB
group1	3	ok	true	0	reg	21.5s	21.5s	0		/tmp/zepto/disk4	29GB
group1	4	ok	true	0	reg	21.5s	21.5s	0		/tmp/zepto/disk1	29GB
group1	4	ok	true	0	reg	20.5s	20.5s	0		/tmp/zepto/disk2	29GB

cost effectiveness



- 18TB диски
- 12TB вложения + 6TB письма
- 20-40 IOPS per disc

Избыточность и сжатие

15PB → x2 → 30PB

30PB → zstd → 15PB

```
* zstd_compress_level = 8
```

Итоги

- BLOB – это не только фото
- доступность (чтение и запись при потере DC)
- надежность (разные схемы репликации)
- эффективность хранения

Исходная задача (уменьшение числа серверов):

- унесли письма, разгрузили индексы
- сократили число серверов 3K → 1.5K
- продолжили работу по оптимизации хранения индексов

Используем

- облако — 80 PB, x1.5
- сниппеты для поиска по почте — 2.4 PB, x2

Показатели

```
Кластер с письмами:
 - маленькие объекты (~40KB)
 - GET avg:   10 msec
 - GET pct90: 30 msec


Облачный кластер x1.5:
 - большие объекты (>1MB)
 - GET avg read:   500 msec
 - GET pct90 read: 2 sec
```

Точки роста

Ручные операции:
- move in
- fix

Random IO:
- если много удалений

Гибридная схема репликации:
- x3 → x2 → x1.5

*enterprise?

[Доклад Андрея Сумина про хранения почтовых вложений]





Спасибо за внимание!



Могилин Виктор

Руководитель группы разработки стораджей в Почте Mail.ru

v.mogilin@corp.mail.ru

Обратная связь и комментарии по докладу по ссылке



